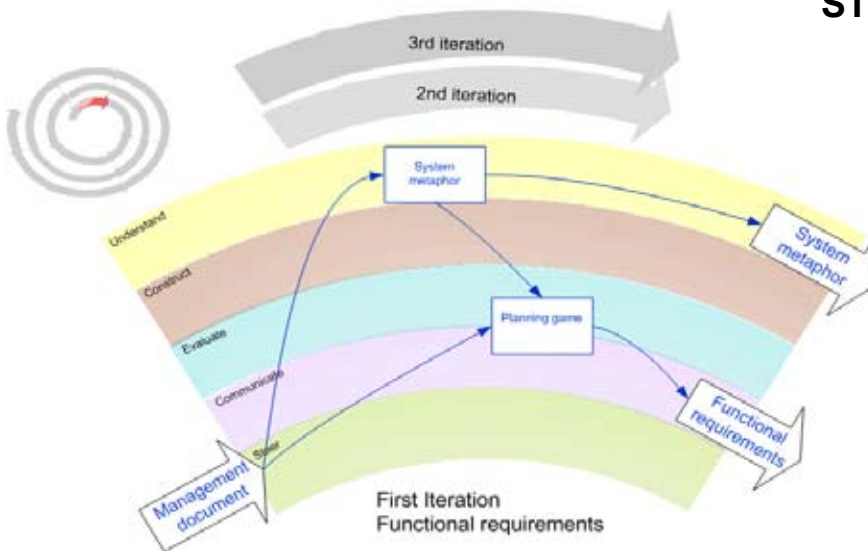


ITERATION 1: FUNCTIONAL REQUIREMENTS

STORIES AND PLANNING GAME



Bite: User stories for the basis of
Time: 2 hours
Input: Notes from first meeting with client, system metaphor
Process evidence: Card stack
Client: Discussion

REQUIREMENTS AND THE PLANNING GAME

The goal of software engineering is to deliver software that does what it is supposed to do, on time and under budget. As we have learned, traditional plan driven methodologies have consistently failed to achieve this. One key factor has been the strategies for the elicitation of requirements. Plan driven waterfall methodologies determine requirements thoroughly at an early stage, then use these requirements to guide the remainder of the process. The problem is: what happens when the requirements change? This might happen when the business itself changes, or different users consider the software and suggest alternative applications, or the client gains a better understanding of the possibilities of the software.

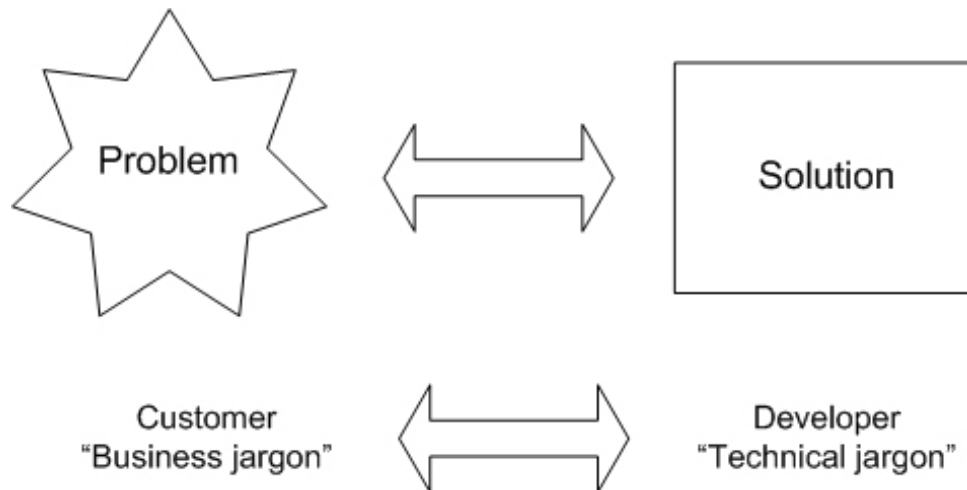
WHAT IS A REQUIREMENT?

A requirement is a statement identifying a capability, physical characteristic, or quality factor that defines a product or process need for which a solution will be pursued.

- **Functional requirements** describe what the system does (capabilities).
Eg, "The system shall enroll students in courses"
- **Non-functional requirements** describe what the system has (physical characteristics).
Includes reliability, quality, safety, maintainability.

LANGUAGE

One of the challenges in the discussion between the customer and the developers is language. The customer is an expert in his business domain, and will use language appropriate to that domain. For example an accountant will use terms such as 'return on investment' (ROI) or 'net present value' (NPV). In the education domain, phrases such as 'path of study' (POS) or 'equivalent full time student' (EFTS) are part of the familiar vocabulary. The developers meanwhile, are familiar with technical terms and can easily confuse the customer



with this jargon.

THE AGILE APPROACH

Agile methods use an approach to determining requirements that can be described as “just in time” requirements analysis. Meetings are held with the customer and/or users in which a best guess is made of what the requirements might be, given the current understanding of the problem. The language used must be understandable to both customer and developer, as the process is the beginning of an ongoing conversation which will continue throughout the development.

Kent Beck, in Extreme Programming (2005) describes a formal requirements gathering process called “The Planning Game”. This is one of the defined Extreme Programming practices. Remember that the key values in Extreme programming are:

- ☐ Communication
- ☐ Simplicity
- ☐ Feedback
- ☐ Courage
- ☐ Respect

The Planning Game applies these values by encouraging early, effective discussion with the whole development team, including the customers. User stories are captured onto index cards and displayed for the whole team to discuss. They are then used to plan the development, including releases and estimation of time. Team members prioritise the cards, discuss constraints and describe tests for the completion of the stories.

Mike Cohn (2004) has developed Beck’s ideas with the application of user stories to agile development.

WHAT IS A USER STORY?

“A user story describes a functionality that will be valuable to either a user or a purchaser of a system or software.” (p4, Cohn, 2004)

The story describes something that the user will do in a single session, for example, “The student will access a project file”, “The student group will have concurrent access to the project files”.

The user stories include three components:

1. The story itself

2. Comments which may clarify details of the story, or raise a point for later discussion
3. Tests to indicating the completion of the story

Jeffries (2001) describes the process of composing the user stories as “Card, Conversation and Confirmation”. The cards are used to capture the initial requirements, which are then discussed among the team and then implemented and tested. This emphasises the stories’ role as a communication tool, to be used in the development process. They are not used to document the process and are not a contractual obligation. Traditional requirements documents were both of these things.

Using this approach allows the team to spread the decision making across the duration of the project. Big picture decisions can be made early, which allows initial planning. Details can be added later as required. Early, large stories are known as ‘epics’.

How big is a story?

Ideally, it describes a task that will require ½ - 10 days coding, assuming an experienced development team. Learning to estimate the size of a story is an important skill for developers. Planning which stories will be included in each iteration is dependent on good time estimations.

Why use story cards?

- ☐ They emphasise verbal rather than written communication
- ☐ They are understandable by all team members
- ☐ They assist effective planning
- ☐ They enable iterative development
- ☐ They encourage deferring detail

(Cohn, 2004)

A good story is:

- ☐ Independent
- ☐ Negotiable
- ☐ Valuable to users or customers
- ☐ Estimatable
- ☐ Small
- ☐ Testable

(Wake, 2003a)

